One Million Shears: Token-Selected Concatenated Linear Transformations

Abstract

A concatenated context-selected linear mapping (CCSLM) need not select arbitrary dense matrices. A particularly economical primitive is the **shear transformation**. Each token in a sequence can select a particular shear, and the selected shears can then be concatenated into a single composite transformation.

This note considers the possibility of a system with approximately **one million tokens**, each contributing one selected shear. The central observation is that a million elementary transformations need not imply a million large matrix multiplications. Shears can be represented and applied extremely cheaply, making a very long sequence of context-selected transformations computationally plausible.

The resulting architecture offers an alternative perspective on token interaction: rather than using attention to repeatedly form weighted combinations of token representations, tokens can collectively specify a transformation of a persistent computational state.

1. The basic construction

Consider a state vector

$$x \in \mathbb{R}^d.$$

Each token t_i selects an elementary shear

$$S_i = S(c_i),$$

where c_i is the context associated with token i.

The complete computation is

$$x_N = S_N S_{N-1} \cdots S_2 S_1 x_0.$$

Thus one million tokens produce

$$x_{10^6} = S_{10^6} \cdots S_2 S_1 x_0.$$

The important point is that the individual S_i do not need to be general matrices.

2. Shears are exceptionally cheap

A two-dimensional shear can be written as

$$S_x(k) = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}.$$

Applied to a vector,

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x + ky \\ y \end{pmatrix}.$$

The transformation therefore requires essentially one multiply and one addition per affected component.

A higher-dimensional elementary shear can similarly modify only one coordinate:

$$x_j \leftarrow x_j + kx_i.$$

Everything else remains unchanged.

Consequently, a token need not carry a dense $d \times d$ matrix. It can specify something as small as

$$(i, j, k),$$

meaning:

> add k times coordinate i to coordinate j.

A million such instructions can therefore represent a very substantial computation without requiring a million dense matrix multiplications.

3. Tokens as transformation instructions

This gives an alternative interpretation of a token sequence.

Instead of viewing each token primarily as a vector that must interact with other token vectors, we can regard a token as contributing an elementary operation:

$$t_i \longrightarrow c_i \longrightarrow S_i.$$

The sequence then becomes something resembling a program:

$$S_1;\ S_2;\ S_3;\ \ldots;\ S_{10^6}.$$

The state is continuously transformed as the sequence is consumed.

This is conceptually different from attention.

Attention asks, approximately:

> Which information from the other tokens should this token incorporate?

A shear-based CCSLM asks:

> What elementary transformation should this token contribute to the evolving computational state?

The distinction is significant.

4. The million-token case

Suppose every one of 10^6 tokens selects one elementary shear.

The naive description might suggest that the resulting computation is enormous because there are a million matrix transformations.

But the matrices are highly structured. If each shear touches only two coordinates, the computational cost can be approximately

$$O(10^6)$$

for a fixed-dimensional state, rather than

$$O(10^6 d^2)$$

for general dense matrix multiplication.

For example, with the operation

$$x_j \leftarrow x_j + k x_i,$$

one token contributes only a small number of arithmetic operations.

Thus the sequence length itself is not necessarily the computational bottleneck.

The more interesting question becomes the dimensionality and structure of the persistent state.

5. The composite transformation

Mathematically, the million operations define one overall linear transformation:

$$M = S_{10^6} \cdots S_1.$$

The final result is simply

$$x_{\text{out}} = M x_0.$$

This creates an interesting distinction between **representation** and **execution**.

The explicit matrix M might be dense and expensive to store.

Yet it may be unnecessary to construct M explicitly.

The system can retain the factorization

$$M = S_{10^6} \cdots S_1$$

and execute the factors sequentially.

This is analogous to representing a complicated operation by a long sequence of simple instructions rather than by explicitly constructing its complete dense representation.

6. Context provides the control pathway

In the CCSLM formulation, the data pathway and control pathway remain separate.

The data state is

$$x.$$

The context is

$$c_i.$$

The context determines the selected operation

$$S(c_i).$$

The computation is then

$$x_{i+1} = S(c_i)x_i.$$

This makes the architecture naturally compatible with the principle:

> **Separate data from control.**

The token sequence can therefore control computation without necessarily being repeatedly mixed into the computational state itself.

7. Comparison with attention

This suggests a potentially useful architectural contrast.

Attention

A conventional attention mechanism creates interactions among token representations. Depending on the implementation, this involves forming queries, keys and values and computing context-dependent weighted combinations.

The computational question is largely:

$$\text{What information should be retrieved from the sequence?}$$

Token-selected shears

The shear formulation instead constructs a sequence of transformations:

$$S_1, S_2, \ldots, S_N.$$

The computational question becomes:

$$\text{What transformation should the sequence collectively perform?}$$

The sequence can therefore function as a **distributed program for transforming a persistent state**.

This does not mean that shears automatically replace attention. Attention is extremely expressive and has important properties that a simple shear sequence does not possess. The point is that the two mechanisms provide fundamentally different ways of exploiting a long token context.

8. Why shears are an interesting primitive

Shears have an especially attractive mathematical property.

An elementary shear has determinant one:

$$\det S_x(k) = 1.$$

It is also exactly invertible:

$$S_x(k)^{-1} = S_x(-k).$$

Consequently, a sequence of shears has an immediate inverse:

$$M^{-1} = S_1^{-1} S_2^{-1} \cdots S_N^{-1}.$$

This provides a route toward long-context computations that do not inherently require information-destroying operations.

Moreover, products of elementary shears can generate substantially more complicated linear transformations. Thus the individual token operations can be extremely simple while their composition becomes highly expressive.

9. Tokens as a computational language

The million-token system can therefore be viewed as having a tiny computational vocabulary.

For example, a token might encode:

$$(i, j, k)$$

with the interpretation

$$x_j \leftarrow x_j + k x_i.$$

A million-token sequence is then effectively a program written in an elementary linear-algebra instruction set.

The remarkable possibility is that the complexity resides primarily in the **composition**, rather than in the individual operations.

This resembles the way a conventional computer can execute a huge program using a very small instruction set.

10. An important limitation

There is an important caveat.

Although the factorized representation

$$S_N \cdots S_1$$

may be cheap to execute, the resulting transformation can become dense.

If the objective is eventually to materialize the complete matrix M, the cost can become

$$O(d^2)$$

in storage and computation.

The proposed architecture therefore works best when the system does **not** require explicit construction of the final matrix.

Instead, the factorized transformation itself becomes the computational representation.

This is potentially an advantage rather than a disadvantage: the system is deliberately representing computation as a sequence of structured transformations.

11. Beyond a fixed shear

A token need not select only the magnitude k.

It could select:

$$(i, j, k)$$

or select from a discrete family,

$$S_1, S_2, \ldots, S_K.$$

Context could determine the selection:

$$c_i \mapsto S_{c_i}.$$

Different layers or stages could have different shear dictionaries.

Thus one part of the system might learn transformations associated with spatial context, another with temporal context, and another with semantic or task context.

There need not be a single global context variable.

12. The million-token thought experiment

The particularly interesting experiment would therefore be something like:

$$10^6 \text{ tokens} \to 10^6 \text{ shear selections} \to \text{one composite transformation}.$$

The experiment would ask how much useful computation can be encoded in this enormous sequence while keeping each individual operation extremely cheap.

Several quantities could then be measured:

- computational cost per token;
- state dimensionality;
- number of distinct shear types;
- effective rank of the resulting transformation;
- sparsity of intermediate transformations;
- degree of reversibility;
- information retained by the persistent state;
- scaling behaviour as sequence length increases;
- comparison with attention at equal computational budgets.

A particularly interesting question is whether performance continues to improve when the sequence becomes extremely long, even though the primitive operation remains tiny.

13. A possible architectural principle

The broader proposal can be summarized as:

$$\boxed{\text{Long context} \to \text{many simple context selections} \to \text{concatenated transformations}}$$

rather than

$$\boxed{\text{Long context} \to \text{many pairwise interactions} \to \text{attention}}$$

The million-token case makes the distinction especially clear.

A token does not necessarily need to retrieve information from every other token. It can instead contribute **one small piece of a global computation**.

The resulting system would treat the token sequence not simply as a collection of representations to be mutually compared, but as a long stream of **context-selected computational instructions**.

That is a potentially useful alternative direction for investigating long-context computation, particularly because elementary shears provide an unusually simple and mathematically well-understood starting point.